

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	11	(Dennie near Shaun).in.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:22
L7	598	allocat\$4 near3 (first adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L8	471	allocat\$4 near3 (second adj (block or segment or portion or section))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L9	224	L7 same L8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L10	2969	("without" or "no" or conceal\$3 or bypass\$4) near (operat\$4 adj system)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L11	2	L9 and L10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L12	5576	(sequential or consecutive) adj2 memory	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L13	5	L9 and L12	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:23
L14	224	L7 same L8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:24

L15	269471	operat\$4 adj system	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:24
L16	81	L15 and L9	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:24
L17	2	11 and 16	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:24
L18	0	1 and 17	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/08/05 12:24


[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

 Terms used [allocation](#) [sequential](#) [memory](#) [bypass](#)

Found 489 of 158,639

 Sort results  
by

 Display  
results

☒ [Save results to a Binder](#)
☒ [Search Tips](#)
☐ [Open results in a new window](#)

 Try an [Advanced Search](#)

 Try this search in [The ACM Guide](#)

Results 1 - 20 of 200

 Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

 Relevance scale ☐ ☐ ☐ ☐ ☐

### 1 [Cache performance of fast-allocating programs](#)

Marcelo J. R. Gonçalves, Andrew W. Appel

 October 1995 **Proceedings of the seventh international conference on Functional programming languages and computer architecture**

 Full text available: [pdf\(1.47 MB\)](#)

 Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)


### 2 [A Large Scale, Homogenous, Fully Distributed Parallel Machine, II](#)

Herbert Sullivan, Theodore R. Bashkow, David Klappholz

 March 1977 **ACM SIGARCH Computer Architecture News , Proceedings of the 4th annual symposium on Computer architecture**, Volume 5 Issue 7

 Full text available: [pdf\(560.05 KB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


The present paper complements the hardware proposal for a fully distributed parallel processor presented in "A Large Scale, Homogeneous, Fully Distributed Parallel Machine, I," by describing a suitable software structure for its operating system management. It is shown that the most basic operating system functions can be performed on a purely local basis, i.e. in such a fashion that identical pieces of the operating systems are concurrently executed by each of the processing el ...

### 3 [Register integration: a simple and efficient implementation of squash reuse](#)

Amir Roth, Gurindar S. Sohi

 December 2000 **Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture**

 Full text available: [pdf\(154.98 KB\)](#)
[ps\(573.81 KB\)](#)

 Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)
[Publisher Site](#)


### 4 [Speculative multithreaded processors](#)

Pedro Marcuello, Antonio González, Jordi Tubella

 July 1998 **Proceedings of the 12th international conference on Supercomputing**

 Full text available: [pdf\(1.24 MB\)](#)

 Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)


**Keywords:** control speculation, data dependence speculation, data speculation, dynamically scheduled processors, multithreaded processors

5 Some issues and strategies in heap management and memory hierarchies

Paul R. Wilson


January 1991: **ACM SIGPLAN Notices**, Volume 26 Issue 3

Full text available:  [pdf\(802.62 KB\)](#) Additional Information: [full citation](#), [citations](#), [index terms](#)

6 Instruction fetching: coping with code bloat

Richard Uhlig, David Nagle, Trevor Mudge, Stuart Sechrest, Joel Emer

May 1995 **ACM SIGARCH Computer Architecture News , Proceedings of the 22nd annual international symposium on Computer architecture**, Volume 23 Issue 2

Full text available:  [pdf\(1.47 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Previous research has shown that the SPEC benchmarks achieve low miss ratios in relatively small instruction caches. This paper presents evidence that current software-development practices produce applications that exhibit substantially higher instruction-cache miss ratios than do the SPEC benchmarks. To represent these trends, we have assembled a collection of applications, called the Instruction Benchmark Suite (IBS), that provides a better test of instruction-cache performance. We discuss th ...

7 Memory Controller Optimizations for Web Servers

Scott Rixner

December 2004 **Proceedings of the 37th annual International Symposium on Microarchitecture**


Full text available:  [pdf\(281.56 KB\)](#) Additional Information: [full citation](#), [abstract](#)

This paper analyzes memory access scheduling and virtual channels as mechanisms to reduce the latency of main memory accesses by the CPU and peripherals in web servers. Despite the address filtering effects of the CPU's cache hierarchy, there is significant locality and bank parallelism in the DRAM access stream of a web server, which includes traffic from the operating system, application, and peripherals. However, a sequential memory controller leaves much of this locality and parallelism unex ...

8 Architectural support for translation table management in large address space machines

Jerry Huck, Jim Hays

May 1993 **ACM SIGARCH Computer Architecture News , Proceedings of the 20th annual international symposium on Computer architecture**, Volume 21 Issue 2


Full text available:  [pdf\(1.34 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Virtual memory page translation tables provide mappings from virtual to physical addresses. When the hardware controlled Translation Lookaside Buffers (TLBs) do not contain a translation, these tables provide the translation. Approaches to the structure and management of these tables vary from full hardware implementations to complete software based algorithms. The size of the virtual address space used by processes is rapidly growing beyond 32 bits of address. As the utilized ad ...

9 Disk cache—miss ratio analysis and design considerations

Alan J. Smith

August 1985 **ACM Transactions on Computer Systems (TOCS)**, Volume 3 Issue 3

Full text available:  [pdf\(3.13 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

The current trend of computer system technology is toward CPUs with rapidly increasing processing power and toward disk drives of rapidly increasing density, but with disk performance increasing very slowly if at all. The implication of these trends is that at some point the processing power of computer systems will be limited by the throughput of the input/output (I/O) system. A solution to this problem, which is described and evaluated in this paper, is disk cache

**10** Register file port requirements of transport triggered architectures

Jan Hoogerbrugge, Henk Corporaal

November 1994 **Proceedings of the 27th annual international symposium on Microarchitecture**

Full text available:  [pdf\(533.24 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Exploitation of large amounts of instruction level parallelism requires a large amount of connectivity between the shared register file and the function units; this connectivity is expensive and increases the cycle time. This paper shows that the new class of transport triggered architectures requires fewer ports on the shared register file than traditional operation triggered architectures. This is achieved by programming data-transport instead of operations. Experiment ...

**11** Performance evaluation of memory consistency models for shared-memory multiprocessors

Kourosh Gharachorloo, Anoop Gupta, John Hennessy

April 1991 **Proceedings of the fourth international conference on Architectural support for programming languages and operating systems**, Volume 19, 25, 26 Issue 2, Special Issue, 4

Full text available:  [pdf\(1.71 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

**12** External memory algorithms and data structures: dealing with massive data

Jeffrey Scott Vitter

June 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 2

Full text available:  [pdf\(628.46 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Data sets in large applications are often too massive to fit completely inside the computers internal memory. The resulting input/output communication (or I/O) between fast internal memory and slower external memory (such as disks) can be a major performance bottleneck. In this article we survey the state of the art in the design and analysis of external memory (or EM) algorithms and data structures, where the goal is to exploit locality in order to reduce the I/O costs. We consider a variety ...


**Keywords:** B-tree, I/O, batched, block, disk, dynamic, extendible hashing, external memory, hierarchical memory, multidimensional access methods, multilevel memory, online, out-of-core, secondary storage, sorting

**13** Supporting dynamic data structures on distributed-memory machines

Anne Rogers, Martin C. Carlisle, John H. Reppy, Laurie J. Hendren

March 1995 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 17 Issue 2

Full text available: Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index](#)

 pdf(2.05 MB)[terms](#), [review](#)


Compiling for distributed-memory machines has been a very active research area in recent years. Much of this work has concentrated on programs that use arrays as their primary data structures. To date, little work has been done to address the problem of supporting programs that use pointer-based dynamic data structures. The techniques developed for supporting SPMD execution of array-based programs rely on the fact that arrays are statically defined and directly addressable. Recursive data s ...

**Keywords:** dynamic data structures

#### 14 [Scalable lock-free dynamic memory allocation](#)

Maged M. Michael

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6

Full text available:  pdf(213.94 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


Dynamic memory allocators (malloc/free) rely on mutual exclusion locks for protecting the consistency of their shared data structures under multithreading. The use of locking has many disadvantages with respect to performance, availability, robustness, and programming flexibility. A lock-free memory allocator guarantees progress regardless of whether some threads are delayed or even killed and regardless of scheduling policies. This paper presents a completely lock-free memory allocator. It uses ...

**Keywords:** async-signal-safe, availability, lock-free, malloc

#### 15 [Architecture and compiler tradeoffs for a long instruction wordprocessor](#)

Robert Cohn, Thomas Gross, Monica Lam

April 1989 **ACM SIGARCH Computer Architecture News , Proceedings of the third international conference on Architectural support for programming languages and operating systems**, Volume 17 Issue 2

Full text available:  pdf(1.48 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A very long instruction word (VLIW) processor exploits parallelism by controlling multiple operations in a single instruction word. This paper describes the architecture and compiler tradeoffs in the design of iWarp, a VLIW single-chip microprocessor developed in a joint project with Intel Corp. The iWarp processor is capable of specifying up to nine operations in an instruction word and has a peak performance of 20 million floating-point operations and 20 million integer operations per sec ...

#### 16 [Cache Memories](#)

Alan Jay Smith

September 1982 **ACM Computing Surveys (CSUR)**, Volume 14 Issue 3

Full text available:  pdf(4.61 MB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

#### 17 [Dynamic memory disambiguation using the memory conflict buffer](#)

David M. Gallagher, William Y. Chen, Scott A. Mahlke, John C. Gyllenhaal, Wen-mei W. Hwu

November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29, 28 Issue 11, 5

Full text available:  pdf(1.31 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

terms

To exploit instruction level parallelism, compilers for VLIW and superscalar processors often employ static code scheduling. However, the available code reordering may be severely restricted due to ambiguous dependences between memory instructions. This paper introduces a simple hardware mechanism, referred to as the memory conflict buffer, which facilitates static code scheduling in the presence of memory store/load dependences. Correct program execution is ensured by the ...

**18** Performance comparison of ILP machines with cycle time evaluation

Tetsuya Hara, Hideki Ando, Chikako Nakanishi, Masao Nakaya

May 1996 **ACM SIGARCH Computer Architecture News , Proceedings of the 23rd annual international symposium on Computer architecture**, Volume 24 Issue 2

Full text available: pdf (1.48 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Many studies have investigated performance improvement through exploiting instruction-level parallelism (ILP) with a particular architecture. Unfortunately, these studies indicate performance improvement using the number of cycles that are required to execute a program, but do not quantitatively estimate the penalty imposed on the cycle time from the architecture. Since the performance of a microprocessor must be measured by its execution time, a cycle time evaluation is required as well as a cy ...

**19** Special session on memory wall: Fighting the memory wall with assisted execution

Michel Dubois

April 2004 **Proceedings of the 1st conference on Computing frontiers**

Full text available: pdf (231.18 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Assisted execution is a form of simultaneous multithreading in which a set of auxiliary "assistant" threads, called *nanothreads*, is attached to each thread of an application. Nanothreads are lightweight threads which run on the same processor as the main (application) thread and help execute the main thread as fast as possible. Nanothreads exploit resources that are idled in the processor because of hazards due to program dependencies and memory access delays. Assisted execution has the po ...

**Keywords:** cache memories, latency tolerance, prefetching, simultaneous multithreading, superscalar processors

**20** The SPEEDES Persistence Framework and the Standard Simulation Architecture

Dr. Jeffrey S. Steinman, Jennifer W. Wong

June 2003 **Proceedings of the seventeenth workshop on Parallel and distributed simulation**

Full text available: pdf (306.39 KB)

Additional Information: [full citation](#), [abstract](#), [index terms](#)[Publisher Site](#)

This paper provides an overview of the SPEEDES persistence framework that is currently used to automate checkpoint/restart for the Joint Simulation System. The persistence framework interfaces are documented in this paper and are proposed standards for the Standard Simulation Architecture. The persistence framework fundamentally keeps track of memory allocations and pointer references within a high-speed internal database linked with applications. With persistence, an object, and the collection of object ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)